## МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЕ РФ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

## “ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Лабораторная работа №4

По дисциплине: «Эвристические методы и алгоритмы» На тему «Теория расписаний»

Выполнил :

Студент группы ВМО31

Оганесьянц К.П.

Проверил:

Кобак В.Г.

Ростов-на-Дону 2022 г.

# Введение

Задачи проектирования и управления в системах, для которых необходимо распределение работы между параллельно работающими разнородными вычислительными устройствами занимают значимое место в теории построения расписаний. Практическая актуальность таких задач определяется существенными возможностями экономии машинного времени и вытекающими функциональными и эксплуатационными преимуществами.

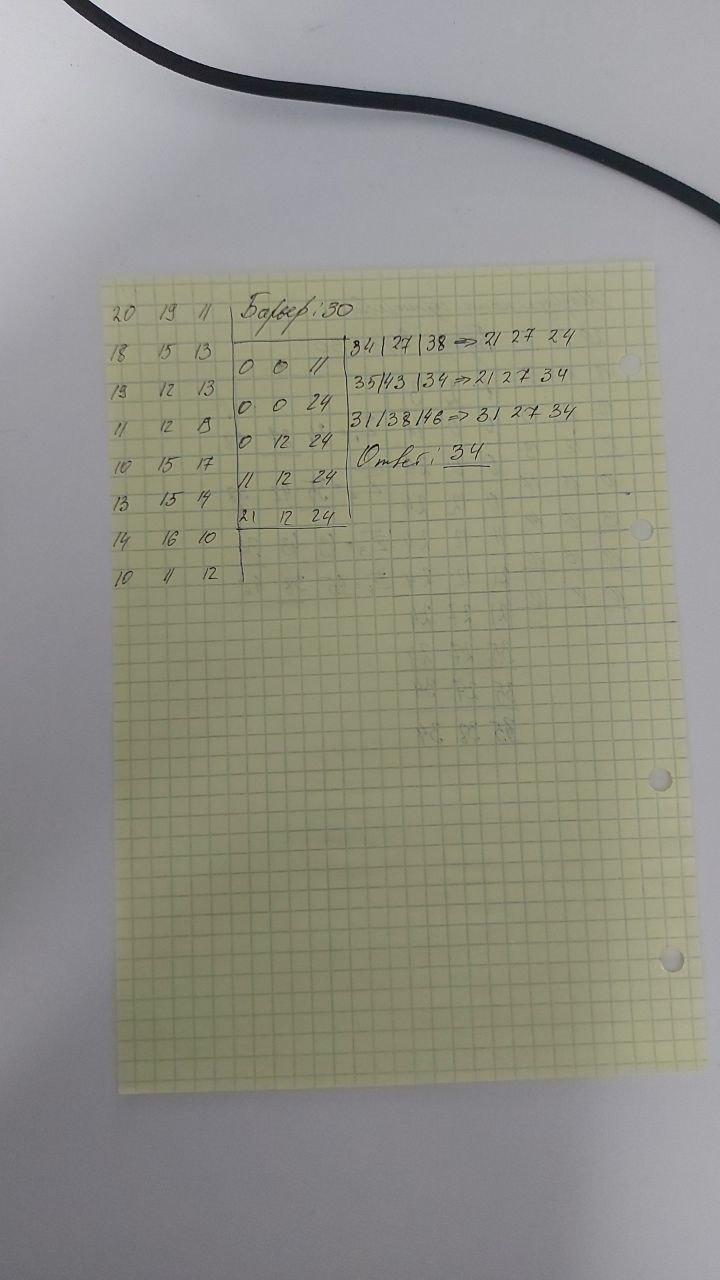
Теоретическая сложность нахождения наилучшего распределения связана с необходимостью решения экстремальных задач комбинаторного типа, требующих больших вычислительных ресурсов, так что эффект от нахождения близкого к оптимальному, с точки зрения времени выполнения, распределения может быть сведен на нет затратами на его получение.

В настоящем руководстве приводятся методы получения расписаний, приводящие к небольшим затратам на вычисление за счет отказа от получения оптимального решения, но в тоже время позволяющие найти приемлемое решение, близкое к оптимальному.

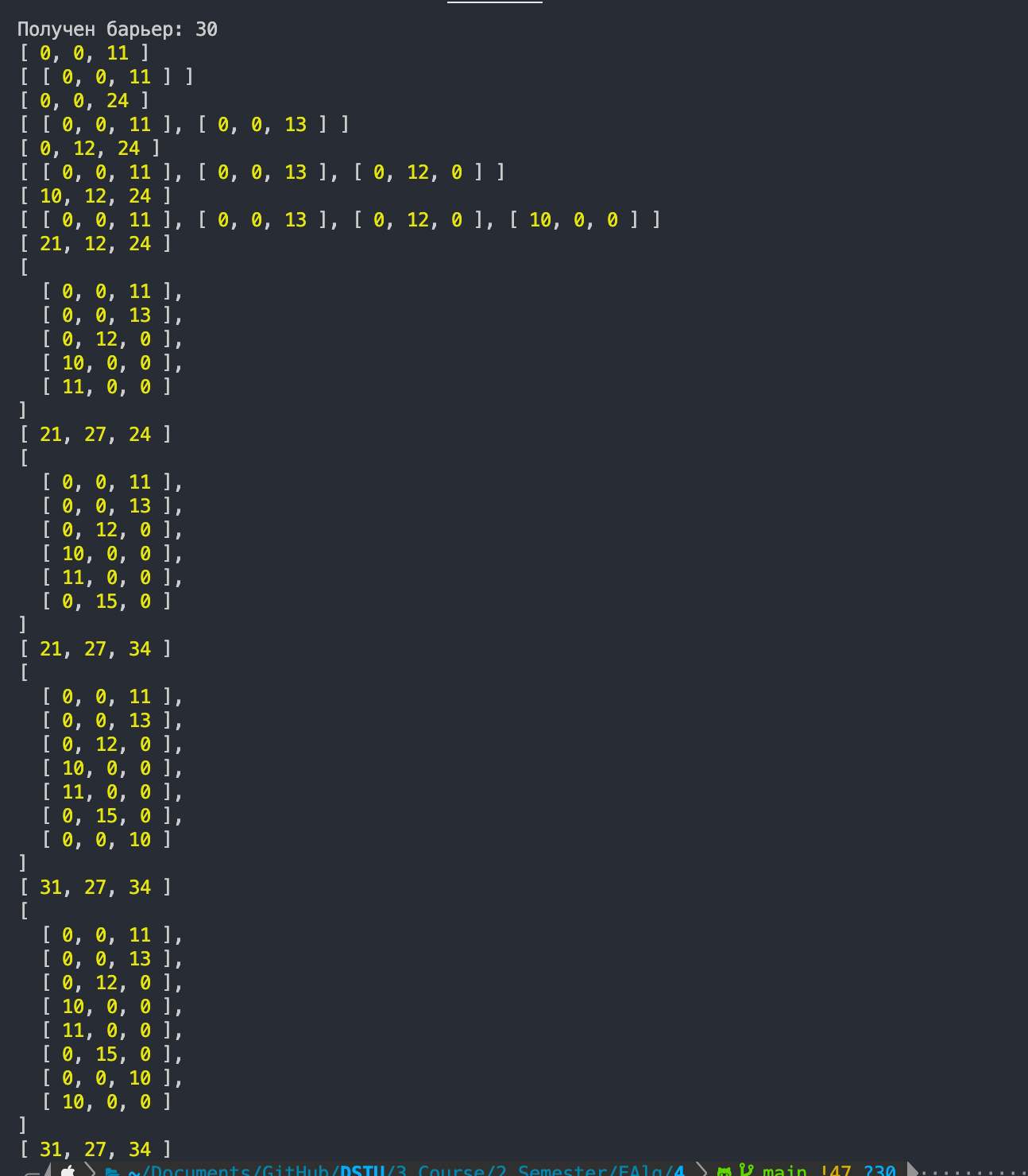
# Постановка задачи

Имеется  независимых работ , которые необходимо распределить на  параллельно работающих разнородных устройств  по критерию .Получение оптимального распределения в такой постановке приводит к громоздким вычислениям, требующим значительного времени машинного счета, поэтому цель – продемонстрировать алгоритмы, с помощью которого можно находить с малыми затратами достаточно приемлемое решение.

1. **Результаты тестирования программы**



Для тестирования программы были заданы параметры.



# Код программы

import readlineSync from 'readline-sync';

const selSort = (array1, array2) => {

const sorted = array.sort(function(a, b){return b.reduce(function(a, b){

return a + b;

}, 0) - a.reduce(function(a, b){

return a + b;

}, 0)});

return sorted;

}

const method = (array, process, result) => {

let minEl;

let minIndex;

for(let x=result.length;x< array.length; x++) {

let extraProcess = []

for(let j=0;j<array[x].length;j++) {

extraProcess.push(array[x][j] + process[j])

}

minEl = extraProcess[0]

minIndex = 0;

for (let i=0;i<extraProcess.length;i++) {

if (minEl > extraProcess[i]){

minEl = extraProcess[i]

minIndex = i;

}

}

process[minIndex] = minEl;

result.push(Array(process.length).fill(0))

result[x][minIndex] = array[x][minIndex]

console.log(process)

console.log(result)

}

return process;

}

export const calculate = (array, process, n) => {

let result = [];

let sum = 0;

let min;

let barier;

for(let q=0;q<array.length;q++) {

min = array[q][0];

for(let r=0;r<array[q].length;r++) {

if (min > array[q][r]){

min = array[q][r]

}

}

sum = sum + min

}

barier = Math.ceil(sum/n)

console.log(`Получен барьер: ${barier}`)

let minEl;

let minIndex;

let check;

for(let r=0;r<= array.length;r++) {

minEl = array[r][0]

minIndex = 0;

for(let e=0;e<=array[r].length;e++) {

if (minEl > array[r][e]) {

minEl = array[r][e]

minIndex = e;

}

}

check = process[minIndex]+minEl

if (check <= barier) {

result.push(Array(process.length).fill(0))

result[r][minIndex] = array[r][minIndex]

process[minIndex] = check

console.log(process)

console.log(result)

}else{

break

}

}

method(array, process, result)

return process

}

function randomIntFromInterval(min, max) { *// min and max included*

return Math.floor(Math.random() \* (max - min + 1) + min)

}

const m = Number(readlineSync.question('Введите m: '));

const n = Number(readlineSync.question("Введите n: "))

const t1 = Number(readlineSync.question("Введите t1: "))

const t2 = Number(readlineSync.question("Введите t2: "))

const variant = Number(readlineSync.question('1-прямая сортировка;2-обратная сортировка;3-случайная'))

*// let array = []*

*// for (let i = 0; i < m; i++) {*

*// let array2 = []*

*// for (let j=0; j < n; j++) {*

*// array2.push(randomIntFromInterval(t1, t2))*

*// }*

*// array.push(array2)*

*// }*

*// console.log(`Начальная матрица: ${array}`);*

let array = [[10,15,17],[19,12,13],[14,16,10],[18,15,13],[11,12,19],[20,19,11],[13,15,14],[10,11,12]]

let arraySum = array.reduce((acc, current) => {

const sum = current.reduce((acc2, el) => {

acc2+=el

return acc2

},0)

acc.push(sum)

return acc

}, [])

switch(variant) {

case 1:

let arraySort = selSort(arraySum, array);

let process = Array(n).fill(0)

let a = calculate(arraySort, process, n)

console.log(a)

break

case 2:

let arraySort2 = selSort(arraySum, array);

let process2 = Array(n).fill(0)

let a2 = calculate(arraySort.reverse(), process2, n)

console.log(a)

break

case 3:

let process3 = Array(n).fill(0)

let a3 = calculate(array, process3, n)

break

}

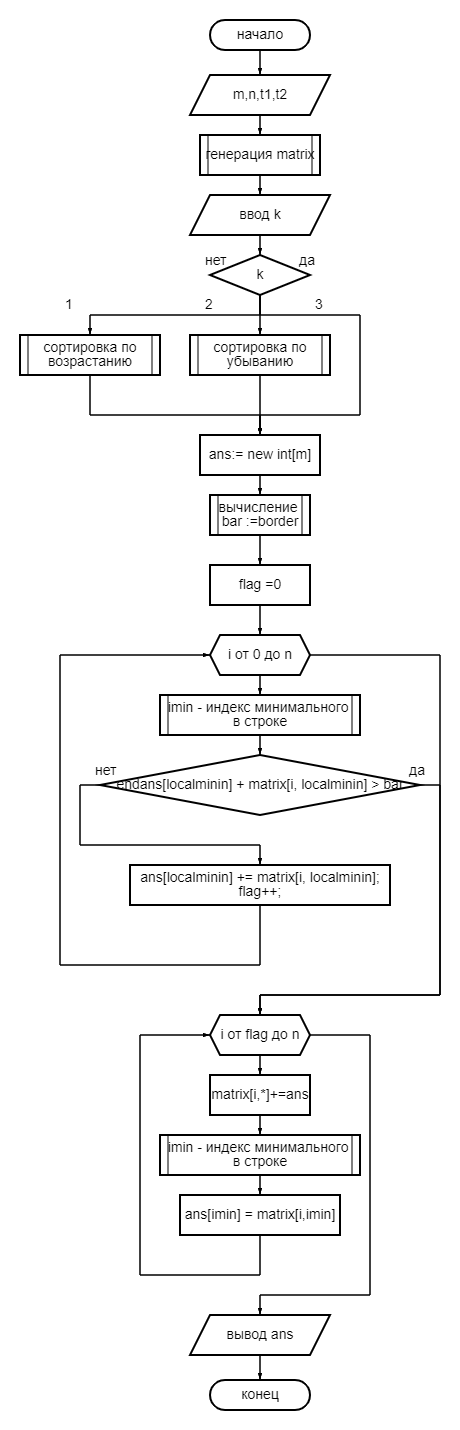
let arraySort = selSort(arraySum, array);

let process = Array(n).fill(0)

let a = calculate(arraySort, process, n)

console.log(a)

# 6 Блок схема

****

1. **Вывод**

Алгоритм построения расписания с методом Плотникова Зверева достаточно эффективен по скорости поиска приемлемого по точности решения.

# Литература

1. Коффман Э.Г. “Теория расписания и вычислительные машины” – M.: “Наука”, 1987
2. Романовский И.В. “Алгоритмы решения экстремальных задач” – М.: “Наука”, 1977
3. Пашкеев С.Д., Минязов Р.И., Могилевский В.Д. “Машинные методы оптимизации в технике связи” – М.: “Связь”, 1976.